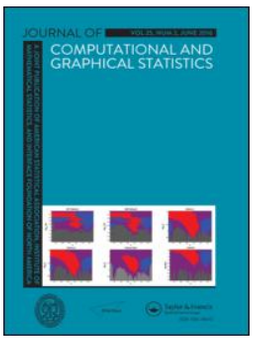

Taylor & Francis
Taylor & Francis Group

Journal of Computational and Graphical Statistics

ISSN: 1061-8600 (Print) 1537-2715 (Online) Journal homepage: http://amstat.tandfonline.com/loi/ucgs20

A Parallel Algorithm for Large-scale Nonconvex Penalized Quantile Regression

Liqun Yu, Nan Lin & Lan Wang

To cite this article: Liqun Yu, Nan Lin & Lan Wang (2017): A Parallel Algorithm for Large-scale Nonconvex Penalized Quantile Regression, Journal of Computational and Graphical Statistics, DOI: 10.1080/10618600.2017.1328366

To link to this article: http://dx.doi.org/10.1080/10618600.2017.1328366

A Parallel Algorithm for Large-scale Nonconvex Penalized Quantile Regression

Liqun Yu*, Nan Lin* and Lan Wang †

Abstract

Penalized quantile regression (PQR) provides a useful tool for analyzing high-dimensional data with heterogeneity. However, its computation is challenging due to the nonsmoothness and (sometimes) the nonconvexity of the objective function. An iterative coordinate descent algorithm (QICD) was recently proposed to solve PQR with nonconvex penalty. The QICD significantly improves the computational speed but requires a double-loop. In this paper, we propose an alternative algorithm based on the alternating direction method of multiplier (ADMM). By writing the PQR into a special ADMM form, we can solve the iterations exactly without using coordinate descent. This results in a new single-loop algorithm, which we refer to as the QPADM algorithm. The QPADM demonstrates favorable performance in both computational speed and statistical accuracy, particularly when the sample size n and/or the number of features p are large.

Key Words: ADMM, nonconvex penalty, parallelization, quantile regression and single-loop algorithm

1 Introduction

Quantile regression, first proposed in the seminar paper (**?**), provides a useful approach to studying the relationship between a response variable and a set of covariates, particularly when the data are heterogeneous. The τth $(0 < \tau < 1)$ conditional quantile of the response variable y given the vector of covariates $\boldsymbol{x} = (x_1, \ldots, x_p)^T$ is defined as $Q_\tau(y|\boldsymbol{x}) = \inf\{t : F_{y|\boldsymbol{x}}(t) \geq \tau\}$, where $F_{y|\boldsymbol{x}}$ is the conditional *c.d.f.* of y

*Liqun Yu is a graduate student and Nan Lin is Associate Professor, Department of Mathematics, Washington University in St. Louis, St. Louis, MO 63130.

†Lan Wang is Professor, School of Statistics, University of Minnesota, Minneapolis, MN 55455.

given $\boldsymbol{x}$. In this paper, we consider estimating a linear quantile regression function $Q_\tau(y|\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{\beta}(\tau)$, where $\boldsymbol{\beta}(\tau) = (\beta_1, \ldots, \beta_p)^T \in \mathbb{R}^p$.

We denote the sample response vector by $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)^T \in \mathbb{R}^n$ and the design matrix by $X = (\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n})^T \in \mathbb{R}^{n\times p}$. In the classical setting where $n > p$, we can estimate $\boldsymbol{\beta}(\tau)$ by

$$\arg\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \sum_{i=1}^{n} \rho_\tau(y_i - \boldsymbol{x_i}^T\boldsymbol{\beta}), \tag{1}$$

where $\rho_\tau(u) = u[\tau - \mathbb{I}(u < 0)]$ for $u \in \mathbb{R}$ is the so-called check loss function with $\mathbb{I}(\cdot)$ being the indicator function. This problem can be formulated as a linear programming program and efficiently computed for moderately large problems, see, for example, the *Quantreg* package (**?**) in R (**?**).

In the high-dimensional setting where $p \gg n$, the estimation problem (**??**) is ill-posed. Under the assumption of sparsity, i.e., most of the covariates are irrelevant for modeling the conditional quantile of interest, we may still obtain reliable estimation via penalization. Specifically, we solve the following penalized optimization problem

$$\hat{\boldsymbol{\beta}}(\tau) = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \Big\{\rho_\tau(\boldsymbol{y} - X\boldsymbol{\beta}) + P_\lambda(\boldsymbol{\beta})\Big\}, \tag{2}$$

where $\rho_\tau(\boldsymbol{y} - X\boldsymbol{\beta}) = \sum_{i=1}^n \rho_\tau(y_i - \boldsymbol{x_i}^T\boldsymbol{\beta})$, $P_\lambda(\cdot)$ is a penalty function and the scalar $\lambda > 0$ is a tunable penalization parameter. For $p \gg n$, the theory for penalized linear quantile regression has been recently investigated by **?** for the Lasso penalty (**?**) and by **?** for the non-convex penalties such as SCAD (**?**) or MCP (**?**).

The nonsmoothness and the possible nonconvexity make problem (**??**) computationally challenging for high-dimensional data. Traditional linear programming algorithms, i.e. the simplex method and the interior point method, are often infeasible in such cases. Recently, **?** proposed an iterative algorithm called QICD to solve (**??**) with nonconvex penalties. The idea is to majorize the objective function in (**??**) by linearizing the penalty, and then minimize the linearized objective function by coordinate descent (CD). This results in a double-loop algorithm where the outer loop is the majorization step and the inner loop is the CD step. The

QICD significantly improves the computational speed and is now implemented in the *QICD* package (**?**) and the *rqPen* package (**?**) in R. In this paper, we propose an alternative algorithm by applying the alternating direction method of multiplier (ADMM). We will demonstrate that by writing the problem (**??**) into a specific parallel ADMM form we can derive a single-loop algorithm that solves (**??**) without using CD. We refer to our algorithm as the QPADM (**Q**uantile regression with **P**arallel **ADM**m). The QPADM is straightforward for parallel implementation on modern computing framework. It is particularly useful when both p and n are large.

The rest of this paper is organized as follows. Section **??** gives a brief introduction to the ADMM algorithm. In Section **??**, we apply a parallel version of the ADMM to solve (**??**) and derive the ADMM updates for the Lasso, SCAD and MCP penalties. The convergence of QPADM for convex and nonconvex penalties is discussed in Section **??**. Section **??** demonstrates the effectiveness of QPADM via numerical simulations. Section **??** concludes the paper.

2 Review of the ADMM

The ADMM was first introduced by **?** and **?**. The algorithm solves the optimization problem

$$\min_{\boldsymbol{x},\boldsymbol{z}} \{f(\boldsymbol{x}) + g(\boldsymbol{z})\} \quad \text{s.t. } A\boldsymbol{x} + B\boldsymbol{z} = \boldsymbol{c}, \tag{3}$$

where $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{z} \in \mathbb{R}^p, A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{m\times p}$, $\boldsymbol{c} \in \mathbb{R}^m$. Problem (**??**) is equivalent to the following augmented form with $\gamma > 0$ being the *augmentation parameter*,

$$\min_{\boldsymbol{x},\boldsymbol{z}} \{f(\boldsymbol{x}) + g(\boldsymbol{z})\} + \frac{\gamma}{2}\|A\boldsymbol{x} + B\boldsymbol{z} - \boldsymbol{c}\|_2^2 \quad \text{s.t. } A\boldsymbol{x} + B\boldsymbol{z} = \boldsymbol{c}. \tag{4}$$

The Lagrangian for problem (**??**) is then

$$L_\gamma\left((\boldsymbol{x},\boldsymbol{z}),\boldsymbol{u}\right) := f(\boldsymbol{x}) + g(\boldsymbol{z}) + \boldsymbol{u}^T(A\boldsymbol{x} + B\boldsymbol{z} - \boldsymbol{c}) + \frac{\gamma}{2}||A\boldsymbol{x} + B\boldsymbol{z} - \boldsymbol{c}||_2^2, \tag{5}$$

where $\boldsymbol{x}$, $\boldsymbol{z}$ together form the primal variable $(\boldsymbol{x},\boldsymbol{z})$ and $\boldsymbol{u}$ is the dual variable. The dual problem of (**??**) is defined as $\max_{\boldsymbol{u}} g(\boldsymbol{u})$ where $g(\boldsymbol{u}) = \max_{(\boldsymbol{x},\boldsymbol{z})} L_\gamma\left((\boldsymbol{x},\boldsymbol{z}),\boldsymbol{u}\right)$ is

the dual function. Under certain conditions, the primal solution can be recovered from the dual problem, i.e., $(\boldsymbol{x}^*, \boldsymbol{z}^*) = \arg\min_{(\boldsymbol{x},\boldsymbol{z})} L_\gamma\left((\boldsymbol{x}, \boldsymbol{z}), \boldsymbol{u}^*\right)$, where $(\boldsymbol{x}^*, \boldsymbol{z}^*)$ is the solution of (**??**) and $\boldsymbol{u}^*$ is the maximum of the dual function. The standard method of multiplier solves (**??**) by iteratively minimizing the Lagrangian over the primal direction and maximizing it over the dual direction. Specifically, it solves

$$\begin{aligned}(\boldsymbol{x}, \boldsymbol{z})^{k+1} &:= \arg\min_{(\boldsymbol{x},\boldsymbol{z})} L_\gamma((\boldsymbol{x}, \boldsymbol{z}), \boldsymbol{u}^k),\\ \boldsymbol{u}^{k+1} &:= \boldsymbol{u}^k + \gamma(A\boldsymbol{x}^{k+1} + B\boldsymbol{z}^{k+1} - \boldsymbol{c}).\end{aligned}$$

at iteration $k+1$. The ADMM applies a slightly different strategy by minimizing the Lagrangian alternatively over $\boldsymbol{x}$ and $\boldsymbol{z}$ directions, i.e., it has the following updates,

$$\begin{aligned}\boldsymbol{x}^{k+1} &:= \arg\min_{\boldsymbol{x}} L_\gamma(\boldsymbol{x}, \boldsymbol{z}^k, \boldsymbol{u}^k),\\ \boldsymbol{z}^{k+1} &:= \arg\min_{\boldsymbol{z}} L_\gamma(\boldsymbol{x}^{k+1}, \boldsymbol{z}, \boldsymbol{u}^k),\\ \boldsymbol{u}^{k+1} &:= \boldsymbol{u}^k + \gamma(A\boldsymbol{x}^{k+1} + B\boldsymbol{z}^{k+1} - \boldsymbol{c}).\end{aligned} \tag{6}$$

Under mild conditions, the ADMM converges to a solution of (**??**).

Many statistical problems can be formulated into (**??**), with functions f and g usually being a loss function and a penalty function, respectively. For example, by introducing a new variable $\boldsymbol{r} = \boldsymbol{y} - X\boldsymbol{\beta}$, problem (**??**) can be written as

$$\hat{\boldsymbol{\beta}}(\tau) = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^p, \boldsymbol{r}\in\mathbb{R}^n} \Big\{\rho_\tau(\boldsymbol{r}) + P_\lambda(\boldsymbol{\beta})\Big\}, \text{ s.t. } \boldsymbol{r} + X\boldsymbol{\beta} = \boldsymbol{y}, \tag{7}$$

which follows form (**??**) with $f(\boldsymbol{x}) = \rho_\tau(\boldsymbol{r})$, $g(\boldsymbol{z}) = P_\lambda(\boldsymbol{\beta})$, $A = I$, $B = X$, and $\boldsymbol{c} = \boldsymbol{y}$.

We refer the readers to **?** for more technical details and a comprehensive overview of the ADMM algorithm.

3 The QPADM

In this section, we apply the ADMM to solve PQR (**??**) with focus on the MCP or SCAD penalty. The algorithm for other penalty functions can be derived similarly.

A direct application of the ADMM to (??) results in the following updates,

$$\begin{aligned}
\boldsymbol{\beta}^{k+1} &:= \arg\min_{\boldsymbol{\beta}} \tfrac{\gamma}{2} \left\|\gamma^{-1}\boldsymbol{u}^k + \boldsymbol{y} - X\boldsymbol{\beta} - \boldsymbol{r}^k\right\|_2^2 + P_\lambda(\boldsymbol{\beta}), \\
\boldsymbol{r}^{k+1} &:= \arg\min_{\boldsymbol{r}} \rho_\tau(\boldsymbol{r}) + \tfrac{\gamma}{2} \left\|\gamma^{-1}\boldsymbol{u}^k + \boldsymbol{y} - X\boldsymbol{\beta}^{k+1} - \boldsymbol{r}\right\|_2^2, \\
\boldsymbol{u}^{k+1} &:= \boldsymbol{u}^k + \gamma(\boldsymbol{y} - X\boldsymbol{\beta}^{k+1} - \boldsymbol{r}^{k+1}).
\end{aligned} \tag{8}$$

The $\boldsymbol{r}$-update in (??) has the following soft-thresholding solution,

$$\boldsymbol{r}^{k+1} := \left[\gamma^{-1}\boldsymbol{u}^k + \boldsymbol{y} - X\boldsymbol{\beta}^{k+1} - \tau\gamma^{-1}\mathbf{1}_n\right]_+ - \left[-\gamma^{-1}\boldsymbol{u}^k - \boldsymbol{y} + X\boldsymbol{\beta}^{k+1} + (\tau-1)\gamma^{-1}\mathbf{1}_n\right]_+. \tag{9}$$

However, the $\boldsymbol{\beta}$-update is expensive since it requires numerical optimization such as CD that results in a double-loop algorithm.

3.1 A parallel implementation that avoids CD

To derive the QPADM algorithm, we first split the data into M blocks as follows,

$$\boldsymbol{y} = \begin{pmatrix} \boldsymbol{y}_1^T & \boldsymbol{y}_2^T & \dots & \boldsymbol{y}_M^T \end{pmatrix}^T, \text{ and correspondingly, } X = \begin{bmatrix} X_1^T & X_2^T & \dots & X_M^T \end{bmatrix}^T,$$

where $\boldsymbol{y}_b \in \mathbb{R}^{n_b}$, $X_b \in \mathbb{R}^{n_b \times p}$, and $\sum_{b=1}^M n_b = n$. Then we rewrite problem (??) into the following equivalent problem by introducing the new variables $\boldsymbol{\beta}_b$'s,

$$\min_{\boldsymbol{r}_b, \boldsymbol{\beta}_b, \boldsymbol{\beta}} \left\{ \sum_{b=1}^M \rho_\tau(\boldsymbol{r}_b) + P_\lambda(\boldsymbol{\beta}) \right\} \quad \text{s.t. } \boldsymbol{y}_b - X_b\boldsymbol{\beta}_b = \boldsymbol{r_b}, \boldsymbol{\beta}_b = \boldsymbol{\beta},\ b = 1, 2, \dots, M. \tag{10}$$

It can be shown that (??) follows form (??), and is solved by the following updates,

$$\begin{aligned}
\boldsymbol{\beta}^{k+1} &:= \arg\min_{\boldsymbol{\beta}} \frac{M\gamma}{2} \left\|\boldsymbol{\beta} - \bar{\boldsymbol{\beta}}^k - \bar{\boldsymbol{\eta}}^k/\gamma\right\|_2^2 + P_\lambda(\boldsymbol{\beta}), \\
\boldsymbol{r}_b^{k+1} &:= \arg\min_{\boldsymbol{r}_b} \rho_\tau(\boldsymbol{r}_b) + \frac{\gamma}{2} ||\boldsymbol{y}_b - X_b\boldsymbol{\beta}_b^k + \boldsymbol{u}_b^k/\gamma - \boldsymbol{r}_b||_2^2, \\
\boldsymbol{\beta}_b^{k+1} &:= (X_b^T X_b + I)^{-1} \left(X_b^T(\boldsymbol{y}_b - \boldsymbol{r}_b^{k+1} + \boldsymbol{u}_b^k/\gamma) - \boldsymbol{\eta}_b^k/\gamma + \boldsymbol{\beta}^{k+1}\right), \\
\boldsymbol{u}_b^{k+1} &:= \boldsymbol{u}_b^k + \gamma(\boldsymbol{y}_b - X_b\boldsymbol{\beta}_b^{k+1} - \boldsymbol{r}_b^{k+1}), \\
\boldsymbol{\eta}_b^{k+1} &:= \boldsymbol{\eta}_b^k + \gamma(\boldsymbol{\beta}_b^{k+1} - \boldsymbol{\beta}^{k+1}),
\end{aligned} \tag{11}$$

where $\bar{\boldsymbol{\beta}}^k = M^{-1}\sum_{b=1}^M \boldsymbol{\beta}_b^k$ and $\bar{\boldsymbol{\eta}}^k = M^{-1}\sum_{b=1}^M \boldsymbol{\eta}_b^k$. We leave the derivation details to supplementary materials.

Compared to (**??**), the formulation (**??**) avoids CD by introducing new variables $\boldsymbol{\beta}_b$. All updates in (**??**) including the $\boldsymbol{\beta}$-updates, now can be solved without iterative methods, as shown at the end of this section. We call (**??**) the QPADM algorithm.

The updates in (**??**) with subscript b depend only on the bth block of the data. When $M > 2$, data blocks $(X_b, \boldsymbol{y}_b)$ can be processed in different computers and hence parallelization can be easily achieved. The parallelization of ADMM was discussed in **?** where the implementation of ADMM on a distributed computing framework called Hadoop (**??**) was presented. We point out that a new generation of Hadoop called Spark (**?**) is faster for iterative computation and hence more suitable for QPADM. We leave the parallel implementation of QPADM to future work.

The matrix inversion in the $\boldsymbol{\beta}_b$-update in (**??**) takes considerable amount of time when p is large. We suggest to use the Woodbury matrix identity (**?**) $(X_b^T X_b + I)^{-1} = I - X_b^T(I + X_b X_b^T)^{-1}X_b$, when p is larger than n_b. This makes the QPADM suitable for the case when both n and p are large. We can choose a split M such that for each b we have $n_b \ll p$, so the $\boldsymbol{\beta}_b$-updates can be implemented efficiently with Woodbury's identity.

Now we show that the updates in (**??**) can be solved without iterative numerical methods. This is clear except for the $\boldsymbol{\beta}$-update. In the following, we derive the solution of the $\boldsymbol{\beta}$-update with the Lasso, SCAD, and MCP penalties. The $\boldsymbol{\beta}$-update for other penalties may be derived in a similar manner.

For the Lasso penalty $P_\lambda(\boldsymbol{\beta}) = \lambda\|\boldsymbol{\beta}\|_1$, the $\boldsymbol{\beta}$-update of QPADM is solved by

$$\boldsymbol{\beta}^{k+1} = (\bar{\boldsymbol{\beta}}^k + \bar{\boldsymbol{\eta}}^k/\gamma - \lambda/(M\gamma)\mathbf{1}_p)_+ - (-\bar{\boldsymbol{\beta}}^k - \bar{\boldsymbol{\eta}}^k/\gamma - \lambda/(M\gamma)\mathbf{1}_p)_-. \tag{12}$$

For the SCAD and MCP penalties, the $\boldsymbol{\beta}$-update is nonconvex. Motivated by the majorization in QICD, at iteration $k+1$, we linearize the penalty $P_\lambda(\boldsymbol{\beta})$ as

$$P_\lambda(\boldsymbol{\beta}) = \sum_{j=1}^p P_\lambda(|\beta_j|) \approx \sum_{j=1}^p P_\lambda(|\beta_j^k|) + P'_\lambda(|\beta_j^k|_+)(|\beta_j| - |\beta_j^k|). \tag{13}$$

Replacing $P_\lambda(\boldsymbol{\beta})$ with the RHS of (**??**) results in

$$\boldsymbol{\beta}^{k+1} := \arg\min_{\boldsymbol{\beta}} \Big\{ \frac{M\gamma}{2} \left\| \boldsymbol{\beta} - \bar{\boldsymbol{\beta}}^k - \bar{\boldsymbol{\eta}}^k/\gamma \right\|_2^2 + \sum_{j=1}^{p} P'_\lambda(|\beta_j^k|_+)|\beta_j| \Big\}. \tag{14}$$

Denoting $\boldsymbol{v}_\lambda^k := \left[P'_\lambda(|\beta_1^k|_+), \ldots, P'_\lambda(|\beta_p^k|_+)\right]^T$, then (**??**) has a closed form solution

$$\boldsymbol{\beta}^{k+1} = (\bar{\boldsymbol{\beta}}^k + \bar{\boldsymbol{\eta}}^k/\gamma - \lambda \boldsymbol{v}_\lambda^k/(M\gamma))_+ - (-\bar{\boldsymbol{\beta}}^k - \bar{\boldsymbol{\eta}}^k/\gamma - \lambda \boldsymbol{v}_\lambda^k/(M\gamma))_-. \tag{15}$$

Neither (**??**) nor (**??**) requires iterative computation.

3.2 Discussion on the convergence of QPADM

The convergence of ADMM for convex problems is well studied in the literature. For example, **?** showed the convergence of ADMM under three assumptions: first, f and g are both convex; second, the global minimum exists for problem (**??**); third, A and B have full column ranks. It is easy to check that these assumptions hold for (**??**) with convex penalties, so the convergence of QPADM is guaranteed for convex PQR. We summarize this as the following theorem,

Theorem 1 *For convex penalties P_λ, the QPADM converges to the solution of (**??**), i.e., $\boldsymbol{\beta}^k$ generated by the QPADM converges to a point $\boldsymbol{\beta}^*$ that solves (**??**).*

While the convergence behavior of ADMM for convex problems is well understood, the convergence of ADMM for nonconvex problems remains unknown. Some recent works, including **?**, **?** and **?**, analyzed the convergence of nonconvex ADMM. Their convergence results crucially rely on the Lipschitz-continuity of the subderivative of the objective function, which is not satisfied in our case. Hence the convergence analysis techniques in the literature cannot be directly applied. Although the behavior of nonconvex ADMM remains largely open, the convergence has been widely observed in practice, see the discussions in **?** and **?**. We observe that the QPADM with either SCAD or MCP penalty has no convergence issue in all examples we have investigated, see Section 4. The theoretical convergence of the nonconvex QPADM, however, still needs further study.

4 Simulation results

We evaluate the computational efficiency, estimation accuracy and feature selection accuracy of the QPADM, and compare it with the QICD. We implemented the QPADM in R and used the R package "QICD" for the QICD simulations. All simulations were conducted on a PC with Core i7 4-core processor and 8GB RAM.

The simulation setup is similar to that in **?**. First, we generate $(\tilde{X}_1, \tilde{X}_2, \ldots, \tilde{X}_p)^T \sim \mathcal{N}(0, \mathbf{\Sigma})$, where $\mathbf{\Sigma}$ is the covariance matrix with elements $\sigma_{ij} = 0.5^{|i-j|}$, $1 \leq i, j \leq p$. Then we set $X_1 = \Phi(\tilde{X}_1)$ and $X_k = \tilde{X}_k$ for $k = 2, 3, \ldots, p$. We consider the following heteroscedastic regression model,

$$Y = X_6 + X_{12} + X_{15} + X_{20} + 0.7X_1\epsilon, \tag{16}$$

where $\epsilon \overset{i.i.d}{\sim} \mathcal{N}(0, 1)$. Three quantile levels were considered: $\tau = 0.3$, 0.5 and 0.7. Notice that the effect of X_1 is only present for $\tau = 0.5$. We chose $(n, p) = (300, 1000)$, $(30000, 1000)$ and $(30000, 100)$ respectively, all with $M = 1$. The simulations were repeated 100 times. Tables 1-3 summarize the results for the SCAD penalty. Results for the MCP penalty are left to the supplementary materials. In the tables, size is the number of nonzero coefficients; P1 is the percentage that $X_6, X_{12}, X_{15}, X_{20}$ were selected; P2 is the percentage that X_1 was selected; AE is the ℓ_1 estimation error; Time measures the running time of the algorithms. Numbers in the parenthesis represent standard deviations.

Following the recent work of **?**, we choose the λ that minimizes

$$\text{HBIC}(\lambda) = \log\Big(\sum_{i=1}^{n} \rho_\tau(y_i - \boldsymbol{x_i}^T\hat{\boldsymbol{\beta}}(\lambda)\Big) + |\mathcal{S}_\lambda|\frac{\log(\log n)}{n}C_n, \tag{17}$$

where $\hat{\boldsymbol{\beta}}(\lambda)$ is the PQR estimator with the tuning parameter λ, $\mathcal{S}_\lambda \equiv \{j : \hat{\beta}_{\lambda,j} \neq 0, 1 \leq j \leq p\}$and $|\mathcal{S}_\lambda|$ is its cardinality, and C_n is a sequence of positive constants diverging to infinity as n increases such that $C_n = O(\log(p))$.

The QPADM performs similarly to QICD in terms of model selection accuracy and estimation accuracy. When n is relatively small (Table 1), QPADM is slightly

slower than the QICD. In fact, QPADM spends most of the time on the matrix inversion while the iterations of QPADM is cheap compared to the iterations of QICD. The computational advantage of QPADM becomes more evident when n gets larger (Tables 2 & 3). This is because, as n increases, the amount of time spent on the matrix inversion ($O(p^3)$) becomes less significant and the time required for the iterations dominates. This is further supported by the results in Table 3.

Method	Quantile	Size	P1	P2	AE	Time (Sec)
QPADM	$\tau = 0.3$	6.17(1.97)	100%	87%	0.051(0.024)	1.57(0.29)
	$\tau = 0.5$	4.42(0.61)	100%	0%	0.040(0.021)	1.65(0.33)
	$\tau = 0.7$	6.21(2.54)	100%	91%	0.049(0.024)	1.68(0.33)
QICD	$\tau = 0.3$	7.33(3.68)	100%	86%	0.049(0.025)	0.91(0.64)
	$\tau = 0.5$	4.19(0.49)	100%	0%	0.039(0.020)	1.57(1.52)
	$\tau = 0.7$	7.17(3.94)	100%	90%	0.051(0.026)	1.33(1.36)

Table 1: Comparison of QPADM and QICD for $(n, p) = (300, 1000)$, SCAD.

Method	Quantile	Size	P1	P2	AE	Time (Sec)
QPADM	$\tau = 0.3$	5.00(0.00)	100%	100%	0.0036(0.0016)	44.97(1.66)
	$\tau = 0.5$	4.01(0.00)	100%	0%	0.0037(0.0019)	46.02(1.71)
	$\tau = 0.7$	5.00(0.00)	100%	100%	0.0039(0.0018)	45.43(1.75)
QICD	$\tau = 0.3$	5.04(0.17)	100%	100%	0.0032(0.0015)	99.83(13.29)
	$\tau = 0.5$	4.10(0.33)	100%	0%	0.0039(0.0014)	125.39(16.35)
	$\tau = 0.7$	5.14(0.27)	100%	100%	0.0030(0.0014)	129.98(17.72)

Table 2: Comparison of QPADM and QICD with $(n, p) = (30000, 1000)$, SCAD.

Method	Quantile	Size	P1	P2	AE	Time (Sec)
QPADM	$\tau = 0.3$	5.00(0.00)	100%	100%	0.0030(0.0011)	3.05(0.63)
	$\tau = 0.5$	4.00(0.00)	100%	0%	0.0029(0.0011)	3.59(0.58)
	$\tau = 0.7$	5.00(0.00)	100%	100%	0.0030(0.0012)	3.98(0.64)
QICD	$\tau = 0.3$	5.04(0.17)	100%	100%	0.0027(0.0011)	11.98(3.64)
	$\tau = 0.5$	4.16(0.37)	100%	0%	0.0026(0.0011)	21.73(11.25)
	$\tau = 0.7$	5.04(0.16)	100%	100%	0.0026(0.0009)	24.99(14.12)

Table 3: Comparison of QPADM and QICD with $(n, p) = (30000, 100)$, SCAD.

Setting $M > 1$ is sometimes necessary when data are too large for a single computer to store and process. We illustrate the advantage of parallelization of

QPADM using the same simulation setup as above for $(n, p) = (30000, 100)$. The block number M is set to 1, 10, and 100 respectively. Denoting the time cost of the matrix inversion $(X_b^T X_b + I)^{-1}$ as T_b^0 and the time at iteration k for the $\boldsymbol{\beta}$-update and updates with subscript b as $T_{\boldsymbol{\beta}}^k$ and T_b^k, respectively, the total time cost after iteration K is calculated as $T^K = \max\{T_1^0, \ldots, T_M^0\} + \sum_{k=1}^K T_{\boldsymbol{\beta}}^k + \sum_{k=1}^K \max\{T_1^k, \ldots, T_M^k\}$. This mimics a real parallel computing framework where the master which conducts the $\boldsymbol{\beta}$-update waits until all workers which conduct updates with subscript b to finish their work before it proceed to the next iteration. The time and estimation accuracy in terms of ℓ_1 estimation error were compared for different M's. The results for $\tau = 0.3$ with SCAD penalty are shown in Figure 1. All other cases follow the same pattern, see the supplementary materials for more details.

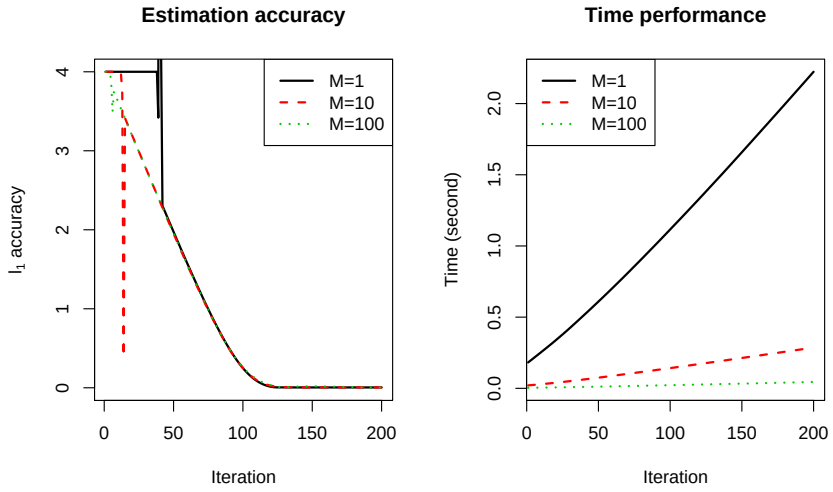

Figure 1: Comparison of QPADM with SCAD penalty for different M values.

As shown in Figure 1, the parallel implementation maintains the estimation accuracy (as measured by the ℓ_1 estimation error) and substantially reduces the computational time. One factor not considered in this simulation is the communication time. When implementing QPADM in a real distributed framework, increasing M will increase the communication overhead. As a result, the computational gain of increasing M will be exceeded by the increase in communication cost at some point.

We point out that our implementation of the QPADM is kept as simple as possi-

ble with no special implementation-level optimization or tuning. The computational advantage of QPADM may not be fully illustrated by the simulations results above.

5 Discussion and conclusion

In this paper, we propose the single-loop QPADM algorithm for PQR. It is computationally advantageous compared to QICD since each of its iteration can be solved without iterative methods. The simulation study showed that the QPADM performs similarly as QICD in terms of statistical accuracy and can be significantly faster when n is large. More importantly, unlike the QICD, the QPADM is a distributed algorithm that can be implemented in distributed framework like Spark. This gives QPADM the ability to solve large scale problems.

To the best of our knowledge, the approach of introducing new variables to the ADMM to avoid iterative methods for the updates is novel in the literature. It can potentially be generalized to other statistical model fitting problems, when the inner-loop of ADMM requires time-consuming iterative algorithms.

Acknowledgement

The authors thank the Editor, Dr. Dianne Cook, and an associate editor for their helpful comments and suggestions that greatly improved the article. Lan Wang's research was partially supported by the National Science Foundation (NSF grant DMS-1512267).

References

Belloni, A., Chernozhukov, V., et al. (2011). ℓ_1-penalized quantile regression in high-dimensional sparse models. *The Annals of Statistics*, 39(1):82–130.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122.

Dean, J. and Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113.

Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96(456):1348–1360.

Gabay, D. and Mercier, B. (1976). A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications*, 2(1):17–40.

Glowinski, R. and Marroco, A. (1975). Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires. *Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique*, 9(2):41–76.

Hong, M., Luo, Z.-Q., and Razaviyayn, M. (2014). Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems. *arXiv preprint arXiv:1410.1390*.

Koenker, R. (2008). quantreg: Quantile regression. r package version 5.11. *URL http://CRAN.R-project.org/package=quantre*.

Koenker, R. and Bassett Jr, G. (1978). Regression quantiles. *Econometrica*, 46(1):33–50.

Lee, E. R., Noh, H., and Park, B. U. (2014). Model selection via Bayesian information criterion for quantile regression models. *Journal of the American Statistical Association*, 109(505):216–229.

Li, G. and Pong, T. K. (2015). Global convergence of splitting methods for nonconvex composite optimization. *SIAM Journal on Optimization*, 25(4):2434–2460.

Mota, J. F., Xavier, J. M., Aguiar, P. M., and Püschel, M. (2011). A proof of convergence for the alternating direction method of multipliers applied to polyhedral-constrained functions. *arXiv preprint arXiv:1112.2295*.

Peng, B. (2016). *QICD: Estimate the Coefficients for Non-Convex Penalized Quantile Regression Model by using QICD Algorithm*. R package version 1.0.1.

Peng, B. and Wang, L. (2015). An iterative coordinate descent algorithm for high-dimensional nonconvex penalized quantile regression. *Journal of Computational and Graphical Statistics*, 64(3):676–694.

R Core Team (2008). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Sherwood, B. and Maidman, A. (2016). *rqPen: Penalized Quantile Regression*. R package version 1.4.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288.

Wang, L., Wu, Y., and Li, R. (2012). Quantile regression for analyzing heterogeneity in ultra-high dimension. *Journal of the American Statistical Association*, 107(497):214–222.

Wang, Y., Yin, W., and Zeng, J. (2015). Global convergence of admm in nonconvex nonsmooth optimization. *arXiv preprint arXiv:1511.06324*.

White, T. (2012). *Hadoop: The Definitive Guide*. O'Reilly Media, Inc.

Woodbury, M. A. (1950). Inverting modified matrices. *Memorandum report*, 42:106.

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I. (2010). Spark: Cluster computing with working sets. In *in Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*.

Zhang, C.-H. et al. (2010). Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*, 38(2):894–942.

ACCEPTED MANUSCRIPT

Supplementary Materials to "A Parallel Algorithm for Large-scale Nonconvex Penalized Quantile Regression"

Liqun Yu*, Nan Lin* and Lan Wang †

In the following supplementary mat

1 Derivation of the QPADM updates

We first demonstrate that problem (9) follows the standard ADMM form (3). Define $A = [A_1\ A_2]$ with

$$A_1 = -\begin{bmatrix} I_p & \dots & I_p & 0_{p\times n}\end{bmatrix}^T \in \mathbb{R}^{(Mp+n)\times p},$$

$$A_2 = \begin{bmatrix} 0_{n\times Mp} & -I_n\end{bmatrix}^T \in \mathbb{R}^{(Mp+n)\times n},$$

and define

$$B = \begin{bmatrix} I_p & & & -X_1^T & & \\ & \ddots & & & \ddots & \\ & & I_p & & & -X_M^T \end{bmatrix}^T \in \mathbb{R}^{(Mp+n)\times Mp},$$

and

$$\boldsymbol{c} = \begin{bmatrix} \mathbf{0} \\ -\boldsymbol{y}\end{bmatrix} \in \mathbb{R}^{Mp+n},\ \boldsymbol{x} = \begin{pmatrix} \boldsymbol{\beta} \\ \boldsymbol{r}\end{pmatrix},\ \boldsymbol{z} = \begin{pmatrix} \boldsymbol{\beta}_1 \\ \vdots \\ \boldsymbol{\beta}_M\end{pmatrix},$$

*Liqun Yu is a graduate student and Nan Lin is Associate Professor, Department of Mathematics, Washington University in St. Louis, St. Louis, MO 63130.

†Lan Wang is Professor, School of Statistics, University of Minnesota, Minneapolis, MN 55455.

Then

$$A\boldsymbol{x} + B\boldsymbol{z} = \begin{bmatrix} \boldsymbol{\beta}_1 - \boldsymbol{\beta}, \\ \vdots \\ \boldsymbol{\beta}_M - \boldsymbol{\beta}, \\ \\ -\boldsymbol{r}_1 - X_1\boldsymbol{\beta}_1, \\ \vdots \\ -\boldsymbol{r}_M - X_M\boldsymbol{\beta}_M \end{bmatrix},$$

where $(\boldsymbol{r}_1^T, \ldots, \boldsymbol{r}_M^T)^T = \boldsymbol{r}$. Then the constraints in (9) can be combined as a single constraint $A\boldsymbol{x} + B\boldsymbol{z} = \boldsymbol{c}$. As a consequence, problem (9) can be written exactly as (3) with $f(\boldsymbol{x}) = \rho_\tau(\boldsymbol{r}) + P_\lambda(\boldsymbol{\beta})$ and $g(\boldsymbol{z}) = 0$.

Next, we show that the standard ADMM updates as in (4) applied to (9) results in the updates (10) in the main article. Applying (4) to (9) with A, B, $\boldsymbol{c}$, $\boldsymbol{x}$, $\boldsymbol{z}$ defined above, and separating the dual variable $\boldsymbol{u}$ corresponding to the constraint $A\boldsymbol{x} + B\boldsymbol{z} = \boldsymbol{c}$ as

$$\boldsymbol{u} = (\boldsymbol{\eta}_1^T, \ldots, \boldsymbol{\eta}_M^T,\ \boldsymbol{u}_1^T, \ldots, \boldsymbol{u}_M^T)^T,$$

where each $\boldsymbol{\eta}_b$ corresponds to the constraint $\boldsymbol{\beta}_b - \boldsymbol{\beta}$ and each $\boldsymbol{u}_b$ corresponds to the constraint $-\boldsymbol{r}_b - X_b\boldsymbol{\beta}_b = -\boldsymbol{y}_b$, we have

$$\begin{aligned}
\begin{pmatrix} \boldsymbol{\beta} \\ \boldsymbol{r} \end{pmatrix}^{k+1} &:= \arg\min_{(\boldsymbol{\beta},\boldsymbol{r})} \sum_{b=1}^M \rho_\tau(\boldsymbol{r}_b) + P_\lambda(\boldsymbol{\beta}) + \sum_{b=1}^M (\boldsymbol{\eta}_b^{k+1})^T(\boldsymbol{\beta}_b^{k+1} - \boldsymbol{\beta}) + \sum_{b=1}^M (\boldsymbol{u}_b^{k+1})^T(\boldsymbol{y}_b - X_b\boldsymbol{\beta}_b^{k+1} - \boldsymbol{r}_b) \\
&\qquad + \sum_{b=1}^M \frac{\gamma}{2}\|\boldsymbol{\beta}_b^{k+1} - \boldsymbol{\beta}\|_2^2 + \sum_{b=1}^M \frac{\gamma}{2}\|\boldsymbol{y}_b - X_b\boldsymbol{\beta}_b^{k+1} - \boldsymbol{r}_b\|_2^2, \\
\begin{pmatrix} \boldsymbol{\beta}_1 \\ \vdots \\ \boldsymbol{\beta}_M \end{pmatrix}^{k+1} &:= \arg\min_{(\boldsymbol{\beta}_1,\ldots,\boldsymbol{\beta}_M)} \sum_{b=1}^M \|\boldsymbol{\beta}_b - \boldsymbol{\beta}^{k+1} + \boldsymbol{\eta}^k/\gamma\|_2^2 + \sum_{b=1}^M \|\boldsymbol{y}_b - X_b\boldsymbol{\beta}_b - \boldsymbol{r}_b^{k+1} - \boldsymbol{u}_b^k/\gamma\|_2^2, \\
\begin{pmatrix} \boldsymbol{\eta}_1 \\ \vdots \\ \boldsymbol{\eta}_M \\ \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_M \end{pmatrix}^{k+1} &:= \begin{pmatrix} \boldsymbol{\eta}_1 \\ \vdots \\ \boldsymbol{\eta}_M \\ \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_M \end{pmatrix}^{k} + \gamma \begin{pmatrix} \boldsymbol{\beta}_1^{k+1} - \boldsymbol{\beta}^{k+1} \\ \vdots \\ \boldsymbol{\beta}_M^{k+1} - \boldsymbol{\beta}^{k+1} \\ \boldsymbol{y}_1 - X_1\boldsymbol{\beta}_1^{k+1} - \boldsymbol{r}_1^{k+1} \\ \vdots \\ \boldsymbol{y}_M - X_M\boldsymbol{\beta}_M^{k+1} - \boldsymbol{r}_M^{k+1} \end{pmatrix}.
\end{aligned}$$

Notice that the $\boldsymbol{\beta}$-update and the $\boldsymbol{r}_b$-updates, and the M $\boldsymbol{\beta}_b$-updates are all separable. Then the updating rules (10) in the main article follow immediately. The closed-form solutions of the $\boldsymbol{\beta}_b$-updates in (10) in the main article follow from the quadratic form of the objective functions as seen in the second update above.

2 Additional simulation results

2.1 Simulation results for the MCP penalty with $M = 1$

In Tables 1-3 of the main article, the performance of the QPADM for model (16) with SCAD penalty were shown. In the following, we show the performance of the QPADM with the MCP penalty.

Method	Quantile	Size	P1	P2	AE	Time (Sec)
QPADM	$\tau = 0.3$	5.80(1.56)	100%	94%	0.048(0.023)	1.65(0.31)
	$\tau = 0.5$	4.31(0.64)	100%	0%	0.036(0.022)	1.54(0.29)
	$\tau = 0.7$	6.80(1.42)	100%	93%	0.043(0.024)	1.67(0.33)
QICD	$\tau = 0.3$	7.56(3.82)	100%	92%	0.050(0.026)	0.99(1.13)
	$\tau = 0.5$	4.24(0.59)	100%	0%	0.040(0.020)	1.51(1.30)
	$\tau = 0.7$	6.80(3.62)	100%	93%	0.049(0.026)	1.46(1.59)

Table 4: Comparison of QPADM and QICD with $n = 300,\ p = 1,000$.

Method	Quantile	Size	P1	P2	AE	Time (Sec)
QPADM	$\tau = 0.3$	5.00(0.00)	100%	100%	0.0040(0.0016)	45.09(1.55)
	$\tau = 0.5$	4.00(0.00)	100%	0%	0.0042(0.0019)	47.16(1.68)
	$\tau = 0.7$	5.00(0.00)	100%	100%	0.0037(0.0017)	44.81(1.57)
QICD	$\tau = 0.3$	5.02(0.14)	100%	100%	0.0031(0.0016)	99.37(11.46)
	$\tau = 0.5$	4.16(0.37)	100%	0%	0.0033(0.0015)	121.47(16.35)
	$\tau = 0.7$	5.08(0.25)	100%	100%	0.0032(0.0014)	118.35(16.17)

Table 5: Comparison of QPADM and QICD with $n = 30,000,\ p = 1,000$.

Method	Quantile	Size	P1	P2	AE	Time (Sec)
QPADM	$\tau = 0.3$	5.00(0.00)	100%	100%	0.0032(0.0011)	3.43(0.56)
	$\tau = 0.5$	4.00(0.00)	100%	0%	0.0031(0.0011)	3.54(0.67)
	$\tau = 0.7$	5.00(0.00)	100%	100%	0.0030(0.0015)	3.42(0.58)
QICD	$\tau = 0.3$	5.06(0.24)	100%	100%	0.0027(0.0011)	12.21(3.33)
	$\tau = 0.5$	4.08(0.27)	100%	0%	0.0026(0.0011)	22.33(11.71)
	$\tau = 0.7$	5.02(0.15)	100%	100%	0.0026(0.0009)	25.45(19.00)

Table 6: Comparison of QPADM and QICD with $n = 30000,\ p = 100$.

3 Parallel QPADM: more results

The main article only showed the performance of parallel QPADM for the SCAD penalty with quantile level $\tau = 0.3$, while the simulation were done with $\tau =$

0.3, 0.5, 0.7 for both the SCAD and MCP penalties. We include the remaining results in Figures 2 - 6.

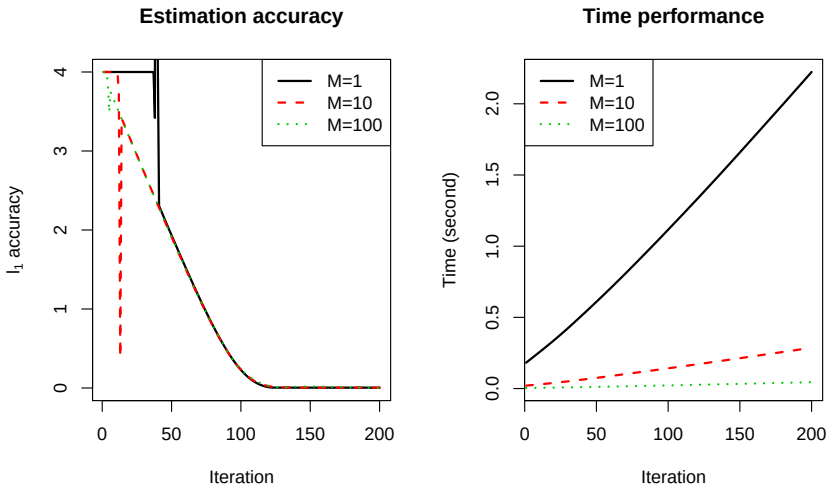

Figure 2: Comparison of QPADM with SCAD penalty for different M values at τ=0.5.

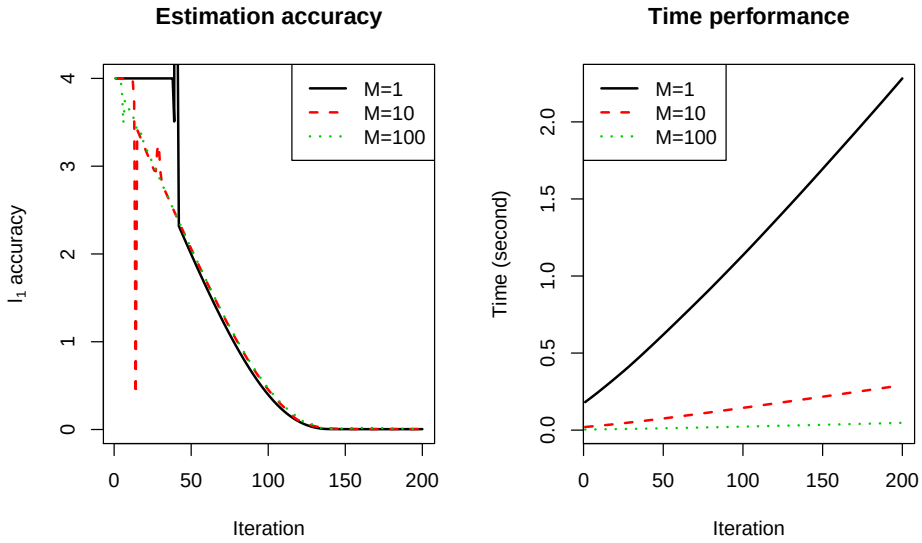

Figure 3: Comparison of QPADM with SCAD penalty for different M values at τ=0.7.

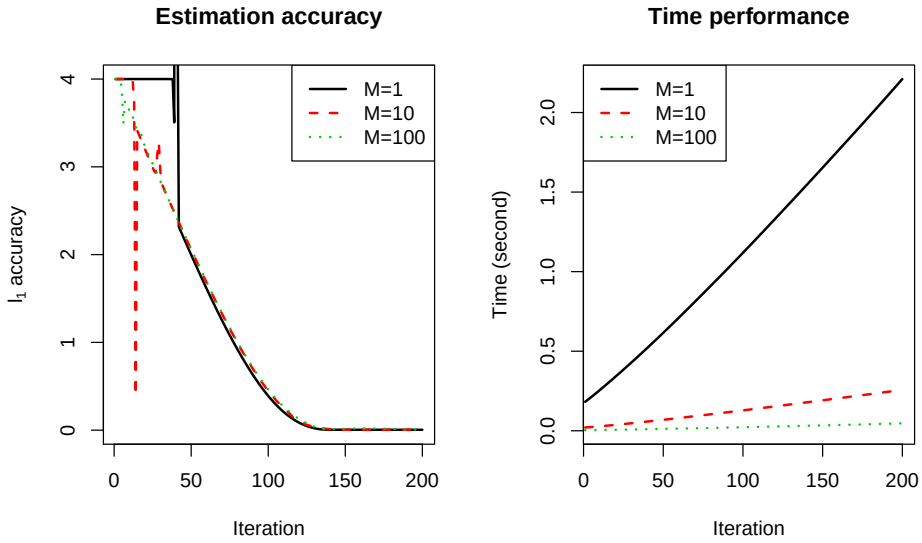

Figure 4: Comparison of QPADM with MCP penalty for different M values at τ=0.3.

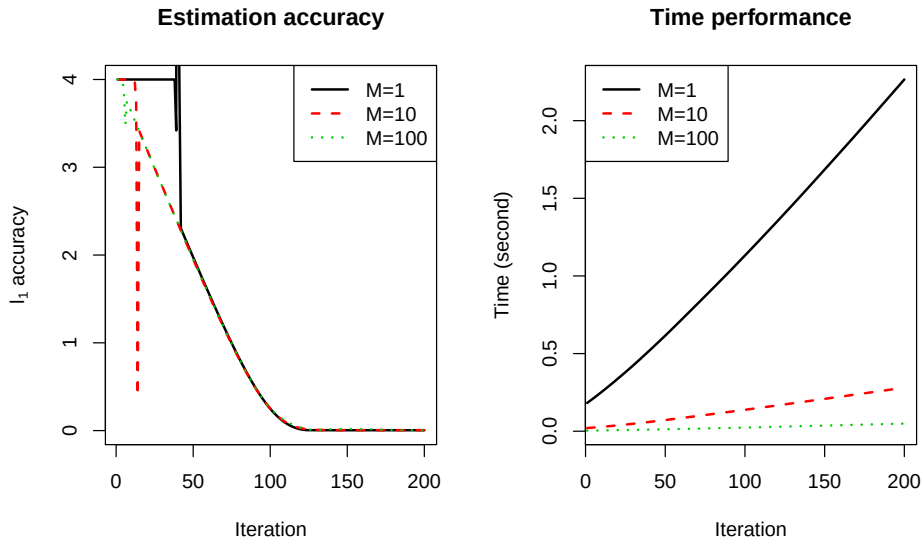

Figure 5: Comparison of QPADM with MCP penalty for different M values at τ=0.5.

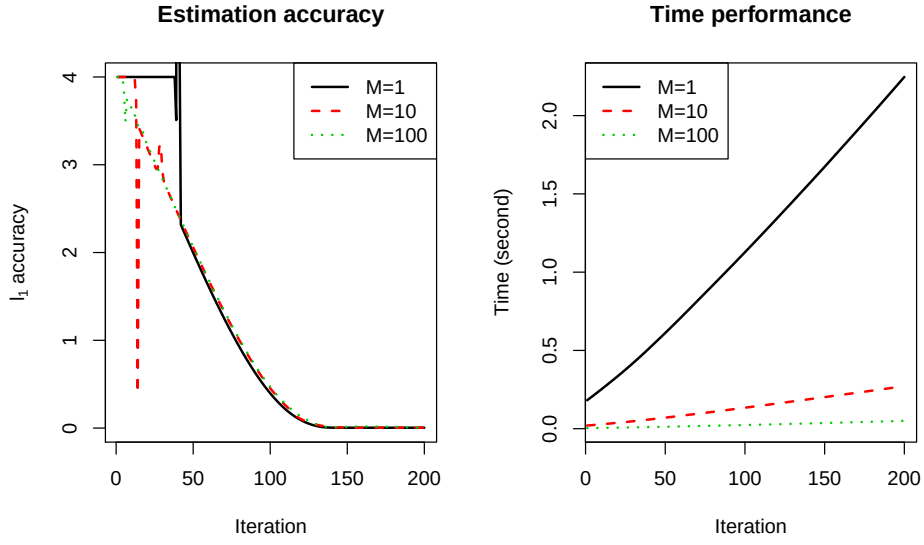

Figure 6: Comparison of QPADM with MCP penalty for different M values at τ=0.7.